Universität Bielefeld
Fakultät für Linguistik und Literaturwissenschaft
31. März 2006
Seminar: Einführung in die Programmierung: Java
Veranstalter: Rüdiger Gleim
Semester: WS 2005/2006

Java-Frontend

für ein XML-annotiertes Lexikon

Daniel Jettka, 1633522
djettka@uni-bielefeld.de

Katharina Jettka, 1719124
katharina.jettka@t-online.de

Inhaltsverzeichnis

Einleitung

Die Idee zu diesem Projekt entstand aus dem Wunsch, ein XML-annotiertes Lexikon als Basis für eine automatische syntaktische Analyse von Texten mithilfe von XSLT zu nutzen. In Form des Morphologiesystems Morphy (siehe Kapitel 1.1) liegt ein Werkzeug vor, mit dessen Hilfe Wörter mit ihren morpho-syntaktischen Eigenschaften verbunden werden können. Da die Ausgabe der gewonnenen Informationen nicht im XML-Format erfolgt, aber dennoch generell gewinnbringend eingesetzt werden kann, müssen die Informationen in ein XML-konformes Format übertragen werden. Um ein auf diese Weise erhaltenes Lexikon weiterentwickeln zu können, haben wir dieses Projekt angelegt. Mit dem Java-Programm sollen im wesentlichen neue Lexikoneinträge eingepflegt oder falsch spezifizierte Einträge gelöscht werden.

Kapitel 1

Das Lexikon

1.1 Format

Das zu verwaltende Lexikon liegt im XML-Format vor. Es wurde mit Hilfe des Morphologiesystems Morphy[1] erstellt. Die Ausgabe der durch Morphy annotierten Wörter erfolgt dabei in SGML-ähnlicher Form. Sie wurde in XML-Notation konvertiert. Um nun eine XML-basierte Verwaltung des auf diese Weise generierten Lexikons zu erreichen, wurde das hier vorgestellte Programm entwickelt.
Im Folgenden soll das Format des Lexikons verdeutlicht werden. Die einzelnen Lexikoneinträge (`<w>`-Elemente) besitzen je nach Wortart verschiedene morpho-syntaktische Informationen. Diese sind in einzelnen Elementen (`<s>`-Elementen) repräsentiert, da ein Wort unterschiedliche syntaktische Verwendungen haben kann. So hat ein Substantiv z.B. Informationen zur Wortart selbst (`w="S"`) und zur Kongruenz (`kgr="..."`) des Wortes in einem bestimmten Verwendungskontext:

```
<w f="Fassung">
    <s w="S" kgr="SI-F-1"/>
    <s w="S" kgr="SI-F-2"/>
    <s w="S" kgr="SI-F-3"/>
    <s w="S" kgr="SI-F-4"/>
</w>
```

Das Wort „Fassung“ ist demnach in vier verschiedenen syntaktischen Formen einsetzbar. Es besitzt in allen vieren die Wortart Substantiv, den Numerus Singular und den Genus Femininum. Nur der Kasus unterscheidet sich an

[1] Morphologiesystem Morphy für Windows95 und NT, Version 1.1; Download: *http://www.wolfganglezius.de/morphy/download.html*

dieser Stelle. Die orthographische Repräsentation ist für alle syntaktischen Verwendungen eines Wortes gleich. Deshalb ist sie als Attribut `f` des übergeordneten `<w>`-Elements realisiert. Dass ein Wort durchaus auch in unterschiedlichen Wortarten vorkommen kann, verdeutlicht das folgende Beispiel:

```
<w f="bequem">
    <s w="V" kgr="SI-IM-"/>
    <s w="A" b="B"/>
</w>
```

Das Wort „bequem“ kann erstens als Imperativ Singular des Verbs „bequemen“ („bequem dich hier her“), zweitens als Adjektiv mit adverbialem Gebrauch („der Sessel ist bequem“) verwendet werden. Anahnd der in einem `<s>`-Element festgelegten Wortart sind die weiteren Attribute desselben Elements bestimmbar. Diese Systematik läßt sich sehr gut für die Repräsentation des Lexikons in Java nutzen (siehe Kapitel 2.1).

1.2 Anforderungen an die Lexikonverwaltung

Die Lexikonverwaltung soll die Pflege des XML-annotierten Lexikons ermöglichen. Es geht zunächst darum, neue Einträge in das Lexikon einzuarbeiten. Dabei müssen einige Besonderheiten beachtet werden. Zuerst muss geprüft werden, ob das neu einzutragende Wort bereits im Lexikon steht. Selbst wenn es schon vorhanden ist, kann es der Fall sein, dass ihm eine neue syntaktische Verwendung zugewiesen werden soll. Dazu müßte der Lexikoneintrag des betreffenden Wortes angezeigt werden, damit der Benutzer beurteilen kann, ob eine neue Verwendung vorliegt oder nicht.
Eine zweite Anwendung ist das Löschen von Lexikoneinträgen. Falsch eingetragene Verwendungen können auf diese Weise korrigiert werden. Desweiteren soll die Lexikonverwaltung einen Gesamtüberblick über das Lexikon ermöglichen. Die Erstellung einer Tabelle auf der Basis der XML-annotierten Informationen wäre hier hilfreich.
Natürlich muss es desweiteren möglich sein, die vorgenommenen Änderungen zu speichern. Dies könnte entweder durch eine direkte Veränderung der XML-Datei, die das eingelesene Lexikon beinhaltet, oder durch Erstellung einer neuen XML-Datei geschehen. Diese Entscheidung sollte dem Benutzer überlassen werden.

Kapitel 2

Java & XML

2.1 Lexikon-Import

Lexikoneinträge in Klassen repräsentieren

Für das in XML annotierte Lexikon wird eine Java-interne Repräsentation durch verschiedene Klassen erstellt. Instanzen dieser Klassen, die im Folgenden erläutert werden, liegen anschließend als Einträge einer HashMap vor.
Nachdem eine XML-Datei, in der ein Lexikon liegt, durch den Benutzer in der Benutzerschnittstelle ausgewählt wurde, beginnt das Programm, diese Datei zu parsen. Dabei wird aus den `<w>`-Elementen eine NodeList erstellt. Die NodeList dient als Parameter der Methode `erstelleWortMap()`. Diese Methode erstellt zunächst ein Objekt der Klasse `HashMap<String,Wort>`. Danach kommt die NodeList zum Einsatz. Für jeden Knoten in der NodeList, das heißt für jedes `<w>`-Element wird die orthographische Repräsentation, die durch das `f`-Attribut angegeben ist, ausgelesen. Die Kindelemente (`<s>`-Elemente) werden in einer NodeList gespeichert. Die beiden Ergebnisse sind die Parameter der Konstruktor-Methode der Klasse Wort. So wird für jedes `<w>`-Element aus der NodeList eine Instanz der Klasse `Wort` erstellt. Die Instanzen werden in der oben erwähnten HashMap gespeichert. Als Key für einen Eintrag fungiert die Orthographie.
Die Konstruktor-Methode von `Wort` erhält, wie schon gesagt, eine NodeList als Parameter. Hierin sind die syntaktischen Eigenschaften eines Wortes in Form von `<s>`-Elementen enthalten. Jedes einzelne `<s>`-Element dient nun wiederum zur Erstellung von Instanzen weiterer Klassen. Die passende Klasse ist abhängig von der Wortart, die im Attribut `w` eines `<s>`-Elements angegeben ist. So teilen sich z.B. die Wortarten Substantiv und Eigenname die Klasse `NuGeKa`. Die Wortart ist ein Indikator dafür, welche weiteren Attribute in einem `<s>`-Element vorhanden sind. Bei Substantiven und Eigennamen

gibt es neben der Wortart (die ohnenhin in jedem `<s>`-Element angegeben ist) nur Informationen über Numerus, Genus und Kasus. Aus dieser Tatsache ist der Name der Klasse `NuGeKa` abgeleitet. Die Parameter der Konstruktor-Methoden der Klassen, die die syntaktischen Eigenschaften eines Wortes repräsentieren, sind jeweils die orthographische Repräsentation des Wortes.

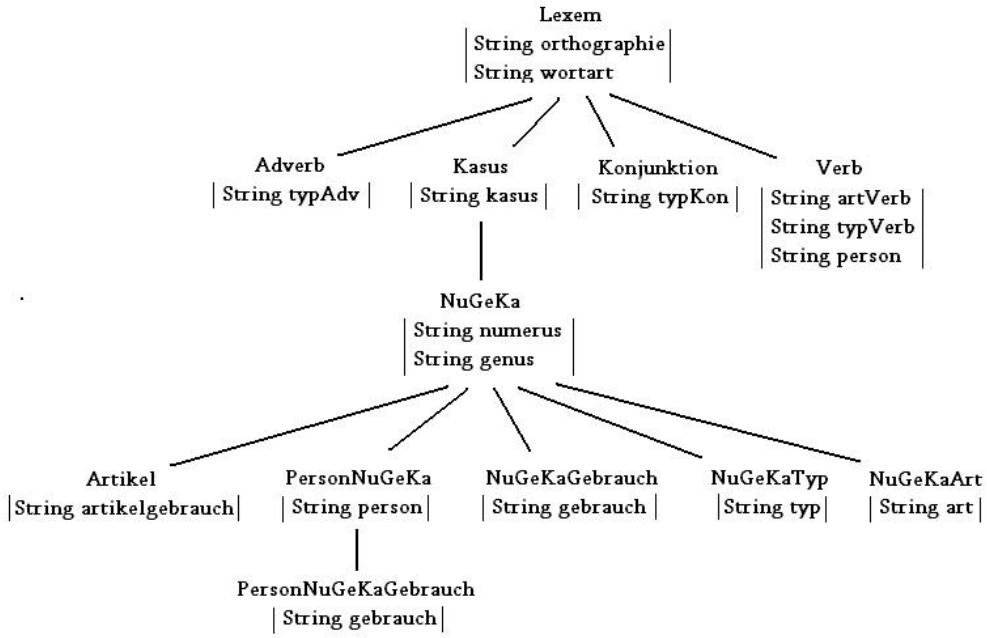

Abbildung 2.1: Klassenhierarchie

2.2 Lexikon-Export

Objekte aus HashMap als XML speichern

Im Kapitel 2.1 wurde erläutert, wie die einzelnen Lexikoneinträge als Objekte der Klasse `Wort` in einer HashMap gespeichert wurden. Die morphosyntaktischen Eigenschaften wurden dabei als Instanzen verschiedener Klassen in einer ArrayList (Attribut des `Wort`-Objekts) gespeichert. Dieser Abschnitt befasst sich nun damit, diese Java-interne Repräsentation in ein XML-annotiertes Lexikon zurückzuführen.
Die dafür zuständige Methode heißt `erstelleXMLAusMap()`. Sie besitzt zwei Parameter: 1. einen String, der den Namen der zu erstellenden XML-Datei darstellt; 2. die HashMap, in der die einzelnen Lexikoneinträge in Form von

Objekten der Klasse `Wort` vorliegen. Diese kann bereits Änderungen enthalten, die der Benutzer u.U. durchgeführt hat. Da die HashMap keine geordneten Einträge enthält ist es zunächst notwendig, sie zu sortieren. Diese Sortierung erfolgt, indem die Keys der HashMap in einen Array geschrieben werden, dessen Einträge anschließend nach dem deutschen Alphabet sortiert werden:

```
Object keys[] = map.keySet().toArray();
Comparator sortDeutsch=Collator.getInstance(Locale.GERMAN);
Arrays.sort(keys, sortDeutsch);
```

Auf diese Weise können die `Wort`-Objekte nun alphabetisch aus der HashMap ausgelesen werden. Nachfolgend wird für jedes in der HashMap gespeicherte Objekt ein `<w>`-Element mit Hilfe der Klasse `PrintWriter` in die XML-Datei geschrieben. Das `f`-Attribut, dessen Wert die orthographische Repräsentation eines Lexikoneintrags angibt, wird unter Rückgriff auf die einzelnen Keys erstellt.
Die syntaktischen Eigenschaften sind, wie schon gesagt, als Attribut der `Wort`-Objekte in einer ArrayList repräsentiert. Im Folgenden werden die einzelnen in der ArrayList vorhandenen Objekte ausgelesen und es wird geprüft, welcher Klasse sie angehören. Anhand der Klassen kann entschieden werden, welche Attribute in den einzelnen Objekten vorhanden sind und analog in die `<s>`-Elemente, die im XML-Format die syntaktischen Eigenschaften beinhalten, aufgenommen werden können. Die Methode `erstelleXMLAusMap()` besitzt keinen Rückgabewert. Das Ergebnis des Funktionsaufrufs ist eine XML-Datei, die das evtl. veränderte Lexikon enthält und deren Name als Parameter mitgeliefert wurde.

Lexikoneinträge in Tabelle visualisieren

Diese Aufgabe ist der zuletzt erläuterten Konvertierung der `Wort`-Objekte aus der HashMap, in der sie gespeichert sind, in ein XML-annotiertes Lexikon sehr ähnlich. Wir haben uns bei der Visualisierung der Lexikoneinträge in einer Tabelle für die Erstellung einer HTML-Datei entschieden. Das HTML-Format ist erstens sehr leicht zu verarbeiten und zweitens ist die Tabelle auch nach der Ausführung des Programms verfügbar.
Die Visualisierung der Lexikoneinträge in einer HTML-Tabelle erfolgt durch die Methode `erstelleTabelleInHTML()`. Diese Methode erstellt eine HTML-Datei, in der in einer Tabelle alle in der als Parameter angegebenen HashMap enthaltenen Lexikoneintrge ausgegeben werden. Als erster Parameter wird der Methode der Name der zu erstellenden HTML-Datei bergeben. An

zweiter Stelle folgt die bereits erwähnte HashMap, in der die einzelnen Wort-Objekte gespeichert sind. Die Ausgabe der einzelnen Lexikoneinträge erfolgt prinzipiell genauso, wie bei der Erstellung des XML-Lexikons (siehe vorangehender Abschnitt). Der grundlegende Unterschied ist lediglich, dass in diesem Fall keine `<w>`- und `<s>`-Elemente aufgebaut werden, sondern die syntaktischen Eigenschaften in Tabellenzellen eingefügt werden.

Kapitel 3

Die Benutzeroberfläche

Zunächst bestand für uns die Frage: AWT oder Swing? Wir haben uns für die Programmierung mit Swing entschieden. Swing ersetzt nicht das AWT, sondern erweitert die AWT-Klassen. Somit ist Swing leistungsfähiger als AWT und bietet mehr Möglichkeiten bei der Gestaltung der Benutzeroberfläche (z.B. interne Frames).
In unserem Programm wird durch eine Anweisung in der main-Methode ein JFrame (projekt) mit dem Titel „Projekt von Daniel & Katharina Jettka“ erstellt. Mit der Anweisung `projekt.setSize(800,300)` wird die Größe des Frame-Fensters bestimmt. Sichtbar gemacht wird das Frame-Fenster durch `projekt.setVisible(true)`. Damit das Frame-Fenster (projekt) durch einen Klick auf das „X“ im Frame-Rahmen geschlossen werden kann, muss das Frame-Fenster auf Ereignisse zum Schliessen reagieren können. Dafür wird folgende Anweisung benötigt:

```
projekt.addWindowListener(new WindowAdapter(){
    public void windowClosing(WindowEvent e){
        System.exit(0);
    }
});
```

Weiterhin benutzen wir interne Frames, das bedeutet Frames, die in dem Hauptfenster dargestellt werden. Um einen neuen internen Frame erstellen zu können, ist der Konstruktor `new JInternalFrame()` nötig. Dieser interne Frame kann dieselben Funktionen haben, wie normale Frames (Ikonifizierbar (`setIconifiable(true)`), Maximierbar (`setMaximizable(true)`), Schliessbar (`setClosable(true)`), ...). Um die Position des internen Frames in dem Hauptfenster zu bestimmen, benutzt man `setBounds(v,w,x,y)`. `v` und `w` sind die Startpositionen des internen Frames im Hauptframe und `x`

und `y` ergeben die Größe des internen Frames. Damit also der interne Frame in der linken oberen Ecke des Hauptfensters beginnt, müssen `v` und `w` 0 sein. Um abschliessend wirklich einen internen Frame zu erstellen, ist es nötig, den internen Frame einem JLayered Pane (= `new JDesktopPane()`) hinzuzufügen. Um den interne Frame wieder unsichtbar zu machen, benötigt man die Anweisung `setVisible(false)`. Damit wird der interne Frame zwar nicht geschlossen, er ist aber für den Benutzer nicht mehr sichtbar.
Von uns benutzte Swing-Komponenten:

JPanel: vergleichbar mit AWT-Panel stellt einen einfachen Container zur Verfügung, der weitere Container und Steuerelemente enthalten kann

JLabel: vergleichbar mit AWT-Label Label werden zur Beschriftung auf grafischen Oberflächen benutzt

JButton: vergleichbar mit AWT-Button Anklickbare Schaltfläche mit Beschriftung

JComboBox: vergleichbar mit AWT-ChoiceButton JComboBox stellt ein Aufklappmenü mit einer Liste von Einträgen zur Verfügung

JTextArea: vergleichbar mit AWT-TextArea Textfläche

JTextField: vergleichbar mit AWT-TextField Textfeld

JTabbedPane: Weiterführung des AWT-Card-Layouts vereinfacht das Wechseln zwischen verschiedenen Layouts

Kapitel 4

Anwendungsbeispiele

4.1 Einen Lexikoneintrag hinzufügen

Bei jedem Start des Java-Programms wird der Benutzer zunächst aufgefordert, eine XML-Datei auszuwählen. Dies dient erstens dazu, verschiedene Lexika (die allerdings alle dasselbe Format haben müssen) einlesen zu können, aber zweitens auch dem Import der Informationen in die Java-interne Klassenrepräsentation. Der Dialog zur Dateiauswahl sieht folgendermaßen aus:

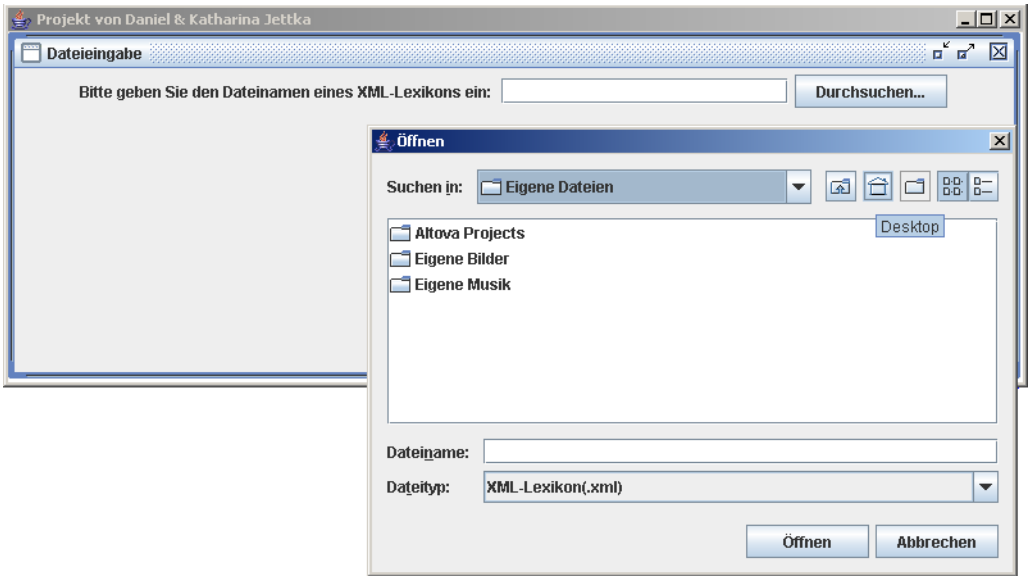

Abbildung 4.1: Einlesen eines XML-Lexikons

Sobald eine XML-Datei eingelesen wurde, werden die dort gespeicherten

Lexikoneinträge in die Java-interne Klassen-Repräsentation überführt. Als nächstes erscheint eine Auswahl über die nächsten möglichen Schritte. Hier kann das Lexikon nach HTML exportiert werden, oder es können bestimmte Operationen mit einzelnen Lexikoneinträgen durchgeführt werden. Dabei kann es z.B. darum gehen, Lexikoneinträge nachzuschauen, neu hinzuzufügen oder aus dem Lexikon zu löschen:

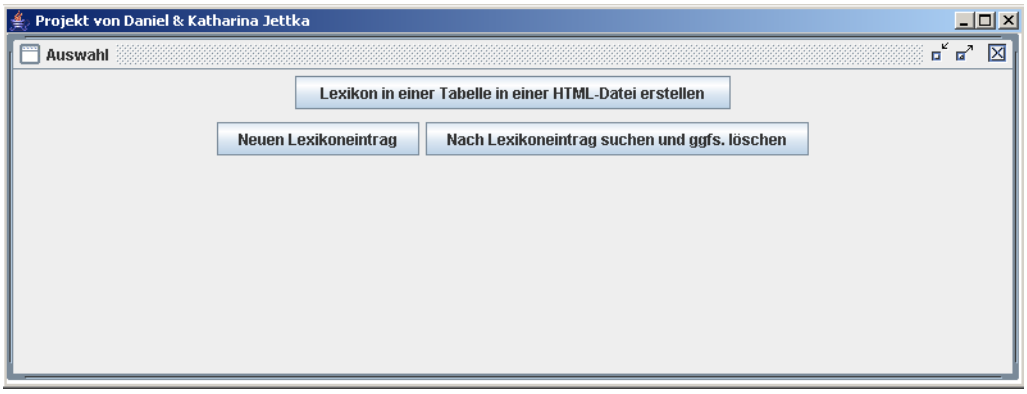

Abbildung 4.2: Auswahl von Verwaltungsaufgaben

Hat der Benutzer die Absicht, einen neuen Lexikoneintrag in das Lexikon vorzunehmen und hat dabei bereits ein bestimmtes Wort im Hinterkopf, so kann er den Menüpunkt „Neuen Lexikoneintrag“ auswählen. Mit einem doppelten Klick auf den Button gelangt man zum Eingabedialog:

Abbildung 4.3: Orthographie eines neuen Lexikoneintrags eingeben

Hier kann das Wort eingetragen werden. Als einfaches Beispiel wählen wir hier einmal den Namen des Flusses „Aa“. Dieser neue Lexikoneintrag kann

im am Ende des Programms gespeicherten neuen XML-Lexikon leicht gefunden werden. Er steht dort als erster Lexikoneintrag.
Das eingetragene Wort muss mit *Enter* bestätigt werden, danach kann der Button „Okay“ geklickt werden. Als nächstes wird dann der Frame zum Eintragen der syntaktischen Eigenschaften des Wortes geöffnet.

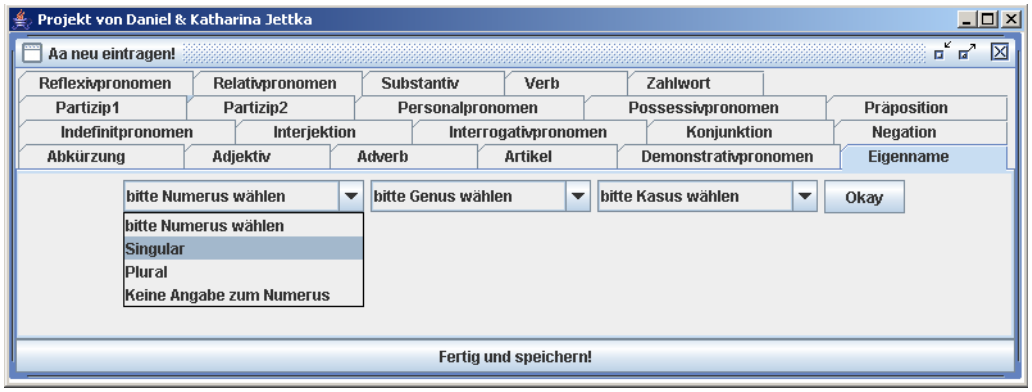

Abbildung 4.4: Orthographie eines neuen Lexikoneintrags eingeben

Nun können die syntaktischen Eigenschaften des Wortes spezifiziert werden. In diesem Beispiel wird für das Wort „Aa“ in der Karteikarten-Perspektive die Wortart Eigenname ausgewählt. Für diese Wortart sind weitere drei syntaktischen Eigenschaften zu bestimmen: Numerus, Genus und Kasus. Alle drei Auswahlboxen müssen bearbeitet werden. Für „Aa“ könnten z.B. der Numerus Singular, das Genus Feminin und der Kasus Nominativ ausgewählt werden. Wurden alle syntaktischen Eigenschaften für die ausgewählte Wortart bestimmt, so kann der Button „Okay“ geklickt werden. Auf Mausklick wird der Lexikoneintrag, falls er nicht in gleicher Form vorhanden war, ins Lexikon aufgenommen. Dem Wort „Aa“ könnten nun noch weitere syntaktische Eigenschaften zugewiesen werden. Dies geschieht nach dem oben beschriebenen Vorgehen. Ist die Angabe abgeschlossen und es sollen keine weiteren syntaktischen Eigenschaften mit dem Wort verbunden werden, klickt man den Button „Fertig und speichern!“. Im Folgenden wird man aufgefordert, eine Datei anzugeben, in der das veränderte Lexikon gespeichert werden soll. Anschließend kann man entscheiden, ob das Lexikon nur im XML-Format oder auch als HTML-Tabelle abgespeichert wird.

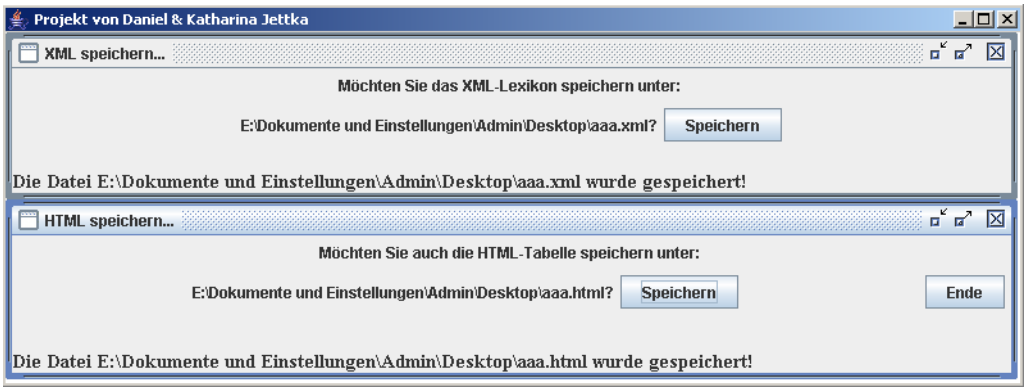

Abbildung 4.5: Orthographie eines neuen Lexikoneintrags eingeben

Das Programm wird nun entweder durch Klick des Buttons „Ende“ oder Schließen des Fensters beendet. In den neu erstellten Dateien lässt sich der neue Eintrag nun nachvollziehen. Das Wort „Aa“ ist nun als erster Lexikoneintrag in der jeweiligen Datei vorhanden.

4.2 Eine Wortfolge überprüfen

Um eine Wortfolge auf hierzu vorhandene bzw. nicht vorhandene Lexikoneinträge zu überprüfen, werden zunächst die gleichen Schritte wie im vorangehenden Beispiel ausgeführt. Wie bei jedem Programmaufruf wird als erstes das zu verwaltende XML-Lexikon eingelesen. Danach gelangt man zur Auswahl der Verwaltungsaufgaben.

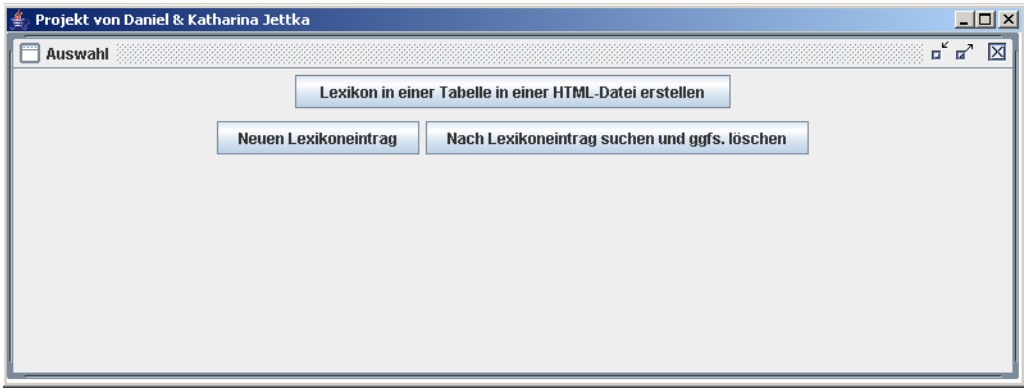

Abbildung 4.6: Auswahl von Verwaltungsaufgaben

Hier wird nun durch einen doppelten Klick des Buttons „Nach Lexikonein-

trag suchen und ggfs. löschen“ ein neuer Dialog geöffnet.

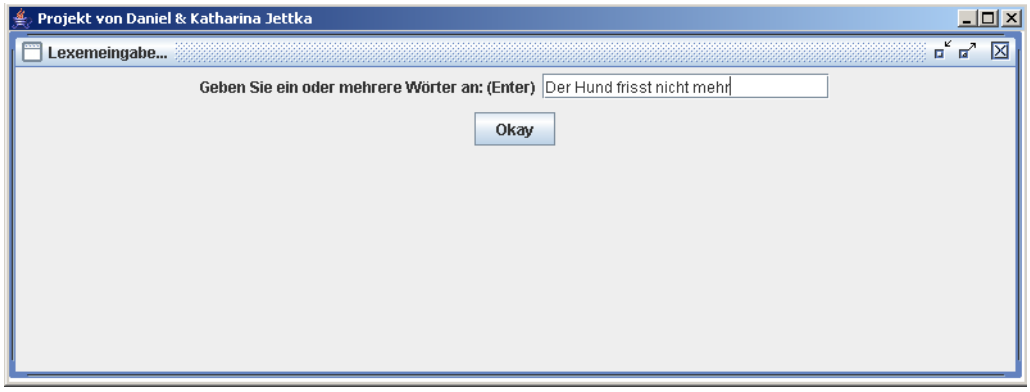

Abbildung 4.7: Lexikoneinträge nachschauen

In diesem kann eine Folge von Wörtern angegeben werden, deren Lexikoneinträge aus dem Lexikon angezeigt werden. Die Eingabe muss zuerst mit *Enter* bestätigt werden. Anschließend wird der Button „Okay“ geklickt. Auf diese Weise werden die Wörter mit den vorhandenen Lexikoneinträgen abgeglichen und falls vorhanden ihre syntaktischen Eigenschaften angezeigt. Von dem verwendeten Beispielsatz ist das Wort „frisst“ nicht im Lexikon vorhanden.

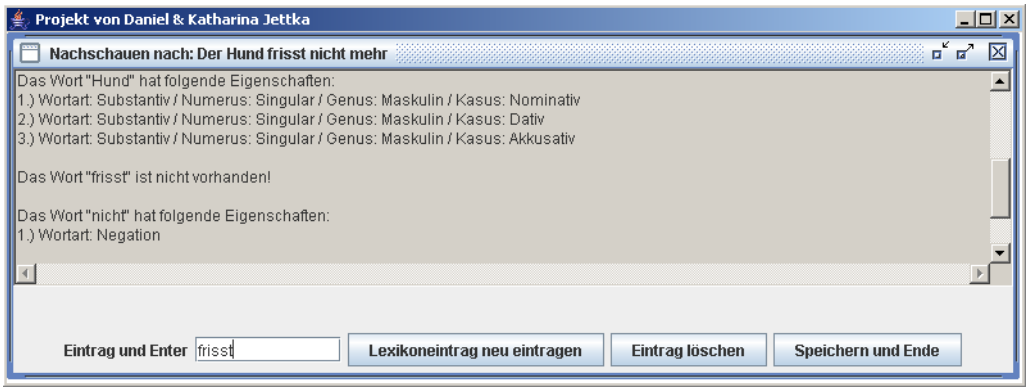

Abbildung 4.8: Lexikoneinträge anzeigen

Mithilfe der Buttons am unteren Rand des Frames lassen sich nun die Lexikoneinträge verwalten. Einerseits können Lexikoneinträge gelöscht werden. Hierzu gibt man im linken Textfeld ein Wort ein und bestätigt dieses mit *Enter*. Durch Klicken des Buttons „Eintrag löschen“ wird der zugehörige

Lexikoneintrag gelöscht. Dies kann mit beliebig vielen Wörtern erfolgen. Andererseits kann im Textfeld ein Wort angegeben werden, das dem Lexikon hinzugefügt werden soll. Dieses müsste wiederum mit *Enter* bestätigt werden. Klickt man danach auf den Button „Lexikoneintrag neu eintragen“, wird der bereits im vorangehenden Abschnitt beschriebene Dialog zur Spezifizierung der syntaktischen Eigenschaften des Wortes geöffnet.